R L. Hollands
ED42

Grid Generation for Turbomachinery Problems

J. Steinhoff
Flow Analysis, Inc.
Tullahoma, TN

K.C. Reddy
Reddy and Associates
Tullahoma, TN

Interim Report of Contract NO. NAS8-36487
for Period: December 1985-October 1986

Submitted to
George C Marshall Space Flight Center
Marshall Space Flight Center, AL 35812

By
Reddy and Associates
Rt.2, Box 253
Tullahoma, TN 37388

November 1986

Table of Contents

GRID GENERATION FOR TURBOMACHINERY PROBLEMS

1. Introduction

In this report we outline the development of a computer code to generate numerical grids for complex internal flow systems such as the fluid flow inside the space shuttle main engine. For computing the viscous compressible flow inside the main engine where the fluid undergoes complex turns through various ducts, it is necessary to discretize the physical space while fulfilling several competing requirements. The topology of the grid structure should depend on the flow solver algorithm being used. Finite difference codes usually require constraints on the grid structure such as separability of the indices for efficient computational procedures, while finite element codes can be implemented with less stringent requirements on the index structure. The grid should provide reasonable resolution of the flow field relative to the total number of nodes being used. This requires providing more grid points in regions of large gradients of flow qualities, such as the viscous zones near solid boundaries. The numerical grid should meet certain smoothness requirements so that the metrics of the curvilinear grid can be computed numerically and these computed metrics are continuous. Unreasonably skewed grid cells and singular points in the grid where the local transformation of the physical space to computational space has very small or very large Jacobians should be avoided if at all possible. Otherwise, such grids will require special handling by the flow solver algorithm and also may give rise to numerical inaccuracies and instabilities.

There are several grid generation techniques and special purpose codes which can generate reasonable grids for simple two dimensional and three dimensional geometries, for both internal and external flows. These techniques fall under two classes: algebraic generators and elliptic generators. Algebraic generators use various interpolation and stretching functions while elliptic generators solve a set of elliptic partial differential equations. While both techniques are effective for simple geometric regions, it is usually difficult to use them to develop a composite grid over a complex internal flow domain. It is particularly difficult to enforce a smooth transition from one zone to another if the physical domain is composed of separate pieces with different geometries. In this report we describe a new blending method to generate smooth grids over complex domains by systematically blending several smooth grids, each of which is developed to conform to different pieces of the domain. In particular, the development of a grid for a 3–D rotor-stator combination in turbomachinery will be outlined.

The first step in developing a grid is to decide upon a grid index structure, which amounts to choosing a global strategy for mapping the physical domain boundaries to the boundaries of the computational domain. The total domain is broken into several simpler overlapping domains. Each subdomain should have a part of its boundary in common with a part of the physical boundary of the total domain. The rest of the boundaries of the

subdomain are not restricted and are chosen according to convenience. A suitable algebraic grid is generated on a subdomain which conforms to the grid spacing and smoothness requirements of the local physical domain boundary. When all the subgrids are generated by simple methods, they will overlap each other and conform to different parts of the physical boundary of the total domain. Suitable weighting functions are then developed to blend all the grids together to generate a smooth grid over the total domain, which conforms to the various boundary requirements. This technique has been developed and tested by Steinhoff for complex two dimensional[1] and three dimensional[2] geometries.

In this report we outline the blending technique for generating a grid for stator-rotor combination at a particular radial section. The computer programs which generate these grids are listed in the Appendices. These codes are capable of generating grids at different cross sections and thus providing three dimensional stator-rotor grids for the turbomachinery of the space shuttle main engine.

2. Blending Methods for Grid Generation

In many cases where a smooth computational grid is required, the boundary of the computational domain can be decomposed into a number of pieces, each of which is fairly simple. We suppose that an adequate grid can be easily generated for each of these pieces, if considered by itself, and describe a method for blending these "elementary" grids into one smooth composite grid which has all of the pieces as its boundary. Examples where this technique can be used include external flow over a entire aircraft, where simple methods exist for generating grids individually over each of the lifting surfaces and the pieces of the body. Other examples include internal flows in turbomachinery where methods exist for generating grids for each element such as a rotor or a stator taken separately. An important feature of the concept is that it can be used recursively. Composite subgrids can first be formed from elementary grids, using the method, then, the same method can be used to form larger composite grids out of these individual subgrids. If algebraic methods are used to form each elementary grid, which can often be done since each piece is simple, then the entire grid generation procedure is algebraic, since the blending is non-iterative and involves no partial differential equation solutions. Accordingly, where applicable, it is a fast method suitable for interactive use. Also, if a partial differential equation is to be solved for some physical quantity and an iterative method is used to solve a set of discrete equations on the grid, which is usually the case, then at each iteration the grid can be quickly re-generated and there is no need to store the entire grid system. This feature can be especially important for large three dimensional problems. This method is very different from other algebraic methods, such as those of Eisemen[3]. Each elementary grid is taken to be previously determined, either by algebraic methods, partial differential equation solution [4], or any other means. These grids can be defined over the entire space, rather than just on surfaces as in "transfinite interpolation" schemes.

An important feature of the method is that it allows the grid designer to use software packages and methods already developed or being developed by others (which can be quite sophisticated and complex) for the elementary grids about each piece of the problem. These can be used as "black boxes," and after each elementary grid is generated the grid designer can blend them together. Also, after a composite, complex grid is generated, if one of the pieces is later modified, only the single new elementary grid need be recomputed and blending into the composite grid.

In this report, grid generation for a stator-rotor combination that exists in the space shuttle main engine is described. Elementary H–grids are formed for the basic profile shapes of the stator and rotor by algebraic methods, they are each blended with proper outer boundary grids, each is given its own camber and rotation and then the composite grid is obtained. Figures of the grids for one radial cross section are shown, but the final computer program will generate three dimensional stator-rotor grids with as many cross sections as necessary.

2.1. The Basic Method

Consider a set of N grids, each spanning the same computational space and approximately the same physical space. For simplicity, we define the computational coordinates to be just the (integer) indices of the grids. Thus, in n dimensions we have an n component vector, $\vec{r}_m(\vec{\ell})(\equiv (x_m(\vec{\ell}), y_m(\vec{\ell}), z_m(\vec{\ell}))$ for $n = 3$) defined on each grid (labeled m) as a function of the indices $\vec{\ell}(\equiv (i,j,k)$ for $n = 3)$. It is important to think of the n components of $\vec{r}_m$ as ordinary smooth functions defined in the computational $(\vec{\ell})$ space. Defining non-negative weighting functions $P^m(\vec{\ell})$, the physical coordinates of the composite grid are then simple weighted sums of those of the elementary grids:

$$\vec{r}_c(\vec{\ell}) = \left[\sum_m P^m(\vec{\ell})\vec{r}_m(\vec{\ell})\right] \Big/ \left[\sum_m P^m(\vec{\ell})\right].$$

The weighting functions are, in general, functions of all of the indices $\vec{\ell}$, and are a function of how close the point $\vec{\ell}$ is to the elementary surface segments. When $\vec{\ell}$ approaches some surface segment, say m_1, then $P^{m_1}(\vec{\ell})$ must approach 1 and all the others P's must approach 0 since there we must have

$$\vec{r}_c(\vec{\ell}) \longrightarrow \vec{r}_{m_1}(\vec{\ell})$$

Some of the "art" of using the method resides in the determination of the functions $P^m(\vec{\ell})$. Since values of $\vec{r}_m(\vec{\ell})$ which define smooth grids are determined separately about each elementary surface, the $P^m(\vec{\ell})$ do not have to do as much work as in an interpolation method where they typically completely determine one of the coordinates. In the next section, it will be seen that very simple functions are sufficient. The main problems arise

when grids must be blended with very different values of $\vec{r}$ in certain regions of $\vec{\ell}$ near an elementary surface. Then, care must be taken that a number of derivatives of $P^m(\vec{\ell})$ are 0 as $\vec{\ell}$ approaches the elementary surface (m_1), in addition to the value of $P^{m_1}(\vec{\ell})$ approaching 1. As more derivatives are made to go to 0, the region in $(\vec{\ell})$ space where $\vec{r}_c(\vec{\ell})$ approaches $\vec{r}_{m_1}(\vec{\ell})$ becomes larger.

2.2. *H*-grid For a Stator-Rotor Combination

To develop *H*-grids for stator-rotor combinations, a sequence of elementary grids are generated and blended together sequentially. *H*-grids are developed for a stator and a rotor separately by the same method. For example, the coordinates of the profile shape of a stator cross section are input into a subroutine to fit a cubic function for the mean camber line of the profile. The cubic is used to define a vertical shearing to approximately straighten the airfoil. After the grid is generated this shearing will be applied in reverse to all the grid points so that the initial airfoil is recovered. The shearing function (of x) is a straight line in front of the leading edge and behind the trailing edge, matching the slope and position of the mean camber line there, and is an interpolating cubic function of x in between. This function is simply subtracted from the initial airfoil coordinates and, after the mappings are complete, added back to each of the grid points to generate the final grid. A basic *H*-grid is generated about the profile shape with camber removed by an algebraic method developed by Pelz and Steinhoff[5]. A detailed study of this mapping was presented in Ref. (5) for a single airfoil, where it was shown that a particular transformation can be used to eliminate the singularity which normally arises at the leading edge in this case. A compressible flow problem was solved on this grid and the solution was shown to be accurate once this singularity was removed. Program I, listed in Appendix I generates basic *H*-grids by this method. A coarse *H*-grid generated by this code is shown in Fig. 1 for a stator profile with camber removed. This grid is blended with a 2nd grid which is a rectangular Cartesian grid with a slit in the middle and has the same topology and number of cells as the basic *H*-grid. The objective is to blend grids 1 and 2 to generate a grid that will approach the *H*-grid near the airfoil and the rectangular grid near the outer boundary. Let (i_1, i_2) and (j_1, j_2) be the interval of i and j indices representing the airfoil surface in the *H*-grid. Let (i_0, i_3) and (j_0, j_3) be the intervals of i and j indices for the outer boundary. The blending is done with a single weight function, $p(\vec{\ell})$:

$$\vec{r}_c(\vec{\ell}) = p(\vec{\ell})\vec{r}_1(\vec{\ell}) - (1 - p(\vec{\ell}))\vec{r}_2(\vec{\ell})$$

where

$$p(\vec{\ell}) = \frac{1}{4}|1 - \cos(\pi\alpha)|\,|1 - \cos(\pi\beta)|$$

$$\alpha(i) = (i - i_0)/(i_1 - i_0) \quad , \quad i_0 \le i < i_1$$

$$= 1 \quad , \quad i_1 \le i < i_2$$

$$= (i_3 - i)/(i_3 - i_2) \quad , \quad i_2 < i < i_3$$

β function is defined similarly. The resulting blended grid is shown in Fig. 2. The outer boundary of the blended grid has the appropriate spacing such that it can obey periodic conditions on the top and bottom and it matches with rotor or stator grids on the sides. The inner H–grid near the airfoil is good for some flow solvers. However, in order to provide better resolution near the leading edge and to simulate the blunt trailing edge, the mesh lines emanating from the leading and trailing edges are split into two lines, thus adding another row of points in the grid. Now the camber is added to all the points and the resulting grid is smoothed with a simple Laplcian type of filter. The computed grid (40 x 18) for a stator is shown in Fig. 3. Fig. 4 shows a fine mesh (40 x 34) for the stator. For a rotor grid, the coordinates are rotated about the leading edge by an angle (32.5° in this case). Figs. 5 and 6 show coarse (40 x 18) and fine (40 x 34) grids for a rotor.

Finally the origin of the rotor coordinates are shifted and the stator-rotor grids are matched to derive a composite grid. Fig. 7 shows the composite stator-rotor grid with coarse spacing (80 x 18), repeated for several blades. The spacing of the exit boundary of the stator grid matches with the spacing of the inflow boundary of the rotor. Thus the two grids always match if they slip past each other in whole increments of the mesh spacing. This feature is necessary for some flow solver algorithms.

Appendix II contains the listing of Program II which takes the output of Program I, namely the basic H–grid for a stator or a rotor profile with camber removed and produces the final stator-rotor grids described above.

3. References

1. J. Steinhoff, "Blending Method for Grid Generation," J. of Computational Physics, vol. 65, No. 2, August 1986, pp. 370–385.

2. J. Steinhoff, "A Recursive Grid Generation Method for Fighter Configurations," presented at International Conference on Numerical Grid Generation in Computational Fluid Dynamics, Landshut, W. Germany, July, 1986.

3. P. R. Eiseman, "Grid Generation for Fluid Mechanics Computations," vol. 17, *Annual Reviews of Fluid Mechanics*, Annual Reviews, Inc., Palo Alto, California, 1985; pp. 487–522.

4. J. F. Thompson, "Elliptic Grid Generation," *Numerical Grid Generation*, J. F. Thompson, ed., North Holland, New York, 1982, pp. 79–106.

5. R. B. Pelz and J. S. Steinhoff, "Multigrid–ADI Solution of the Transonic Full Potential Equation for Airfoils Mapped to Slits," *Computers in Flow Predictions and Fluid Dynamics Experiments*, Proceedings of the ASME Winter Meeting, Washington, D.C., November, 1981, pp. 27–34.

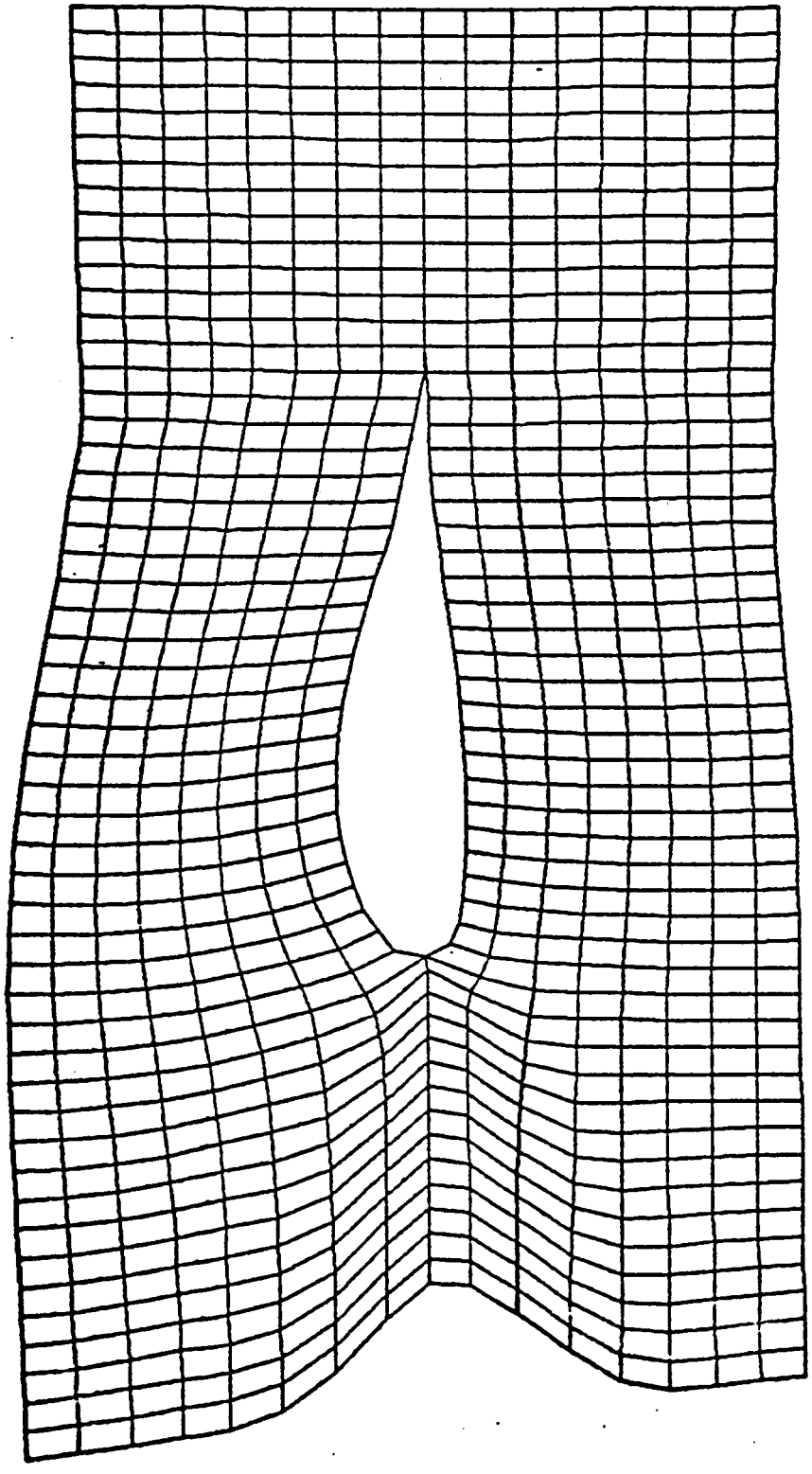

Fig. 1 Basic H - Grid for a Stator Profile with Camber Removed

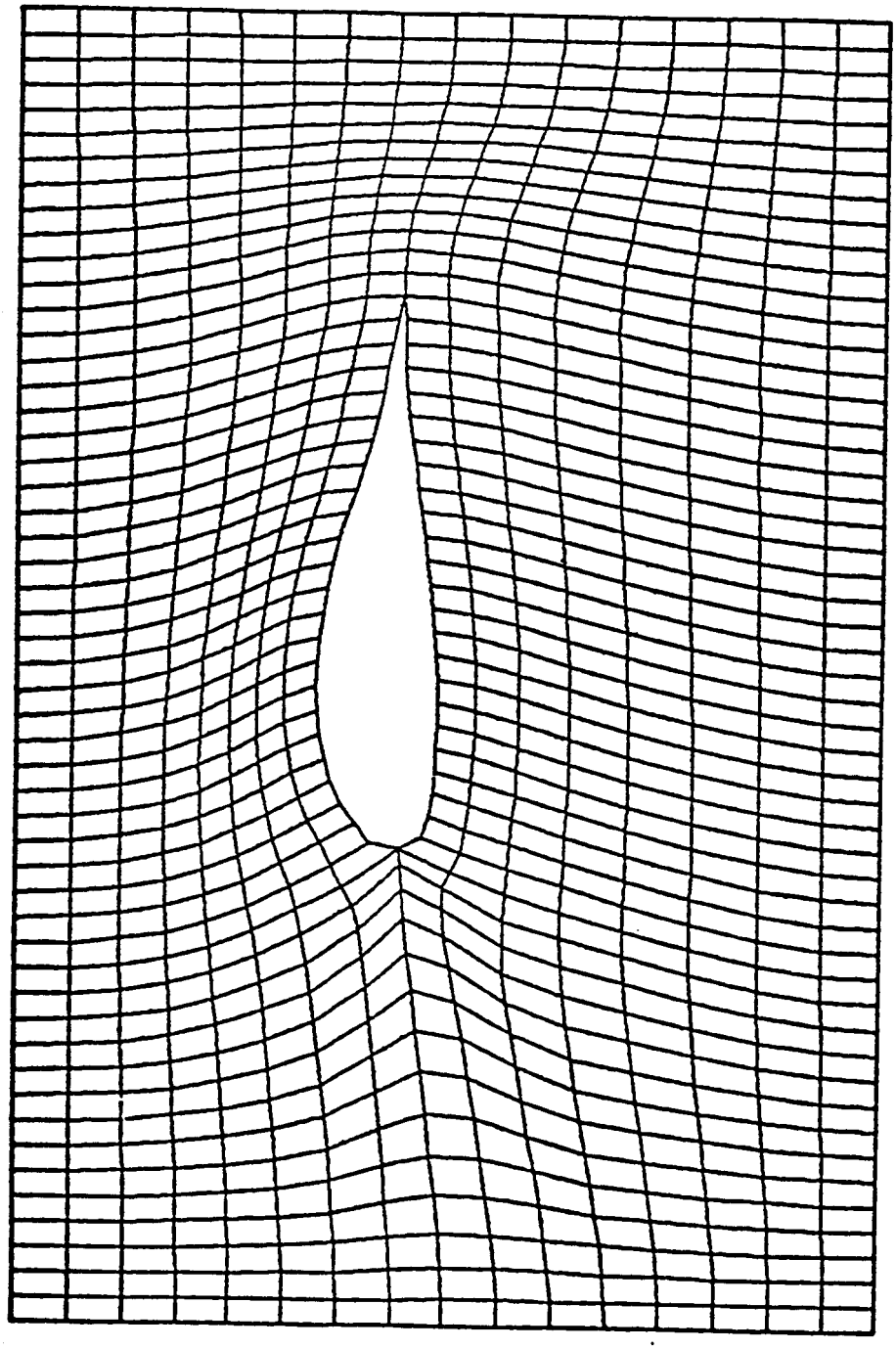

Fig. 2 H - Grid blended with an outer rectangular grid

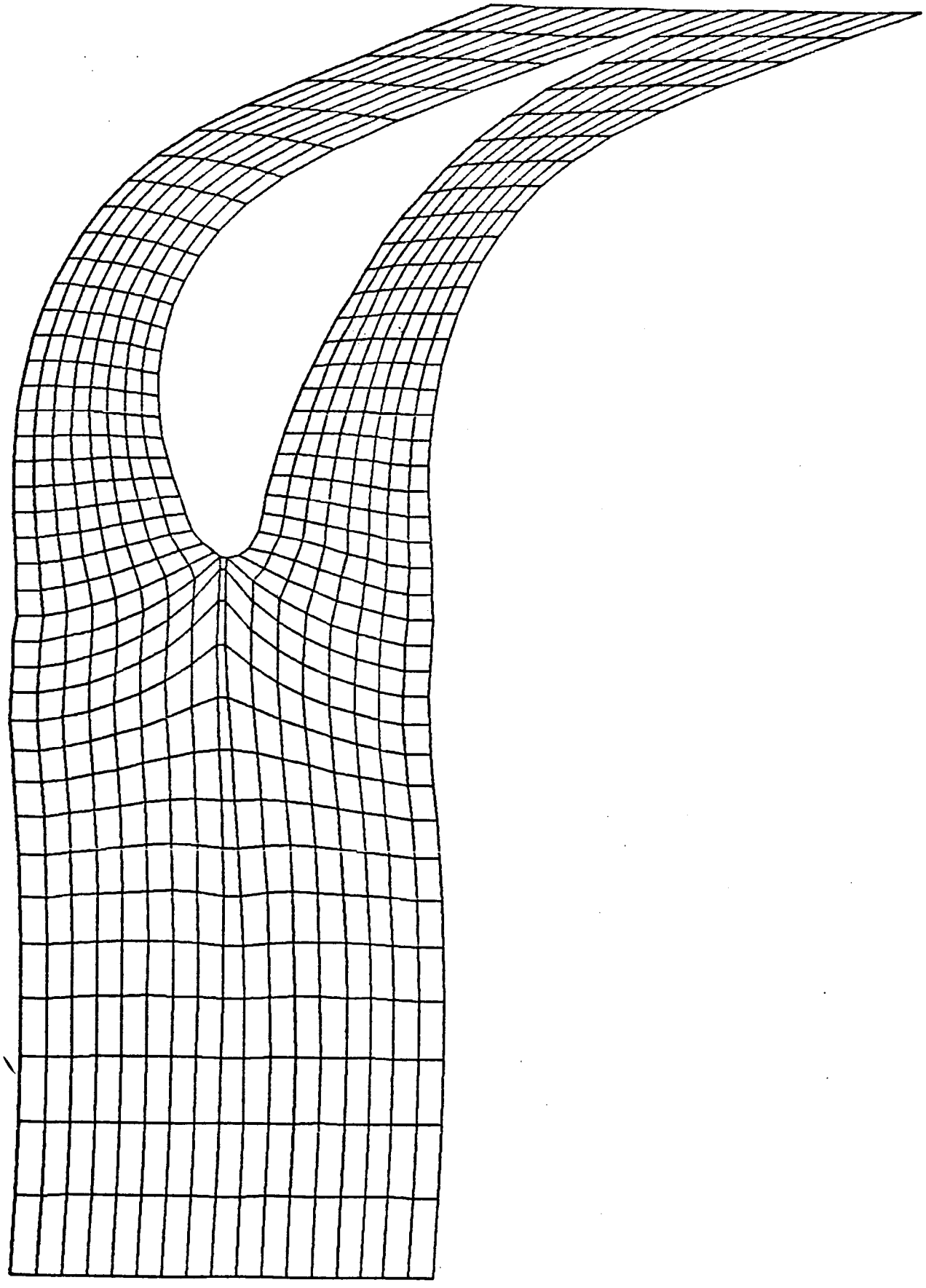

Fig. 3 Stator Grid - Coarse Mesh (40 x 18)

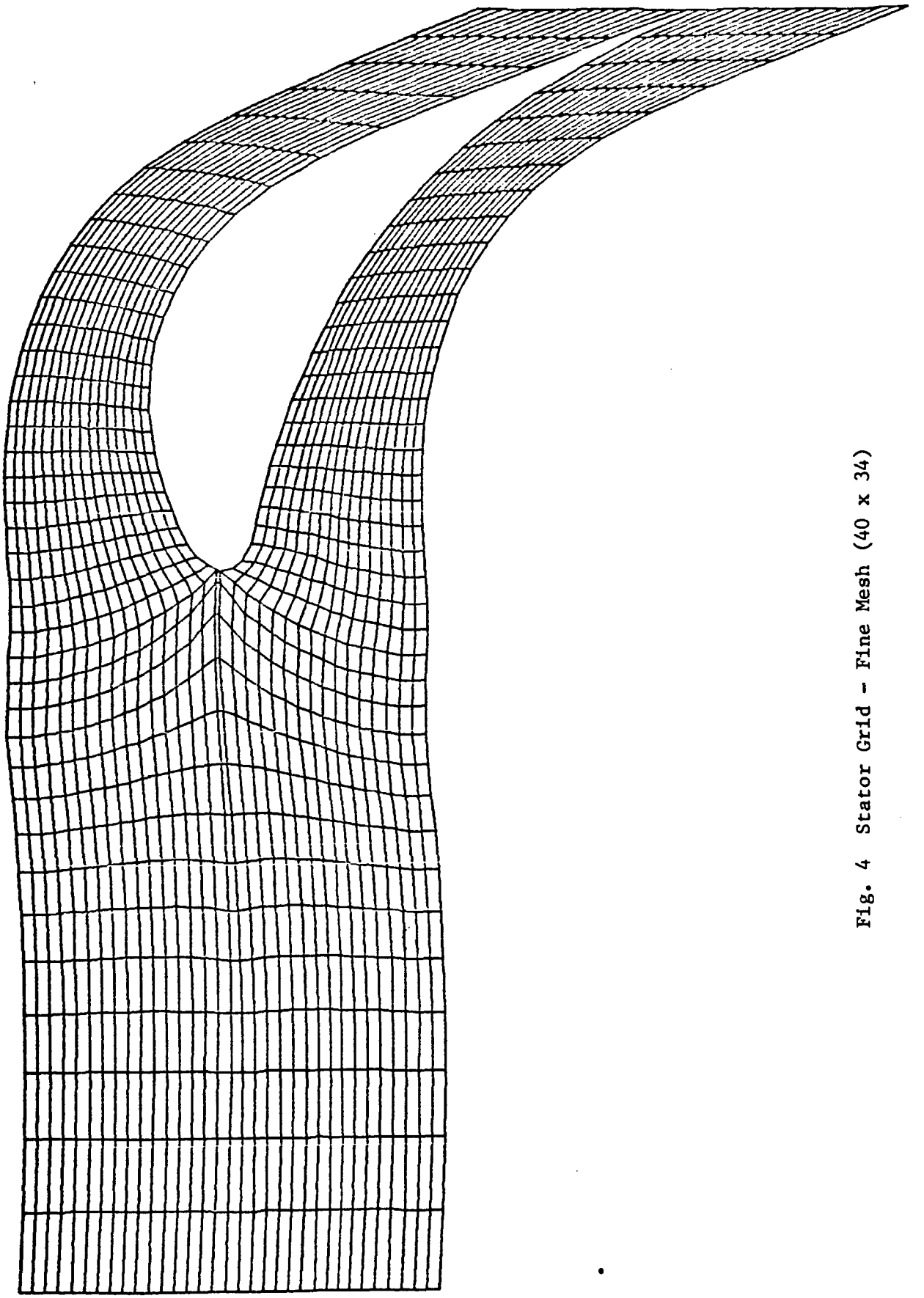

Fig. 4 Stator Grid - Fine Mesh (40 x 34)

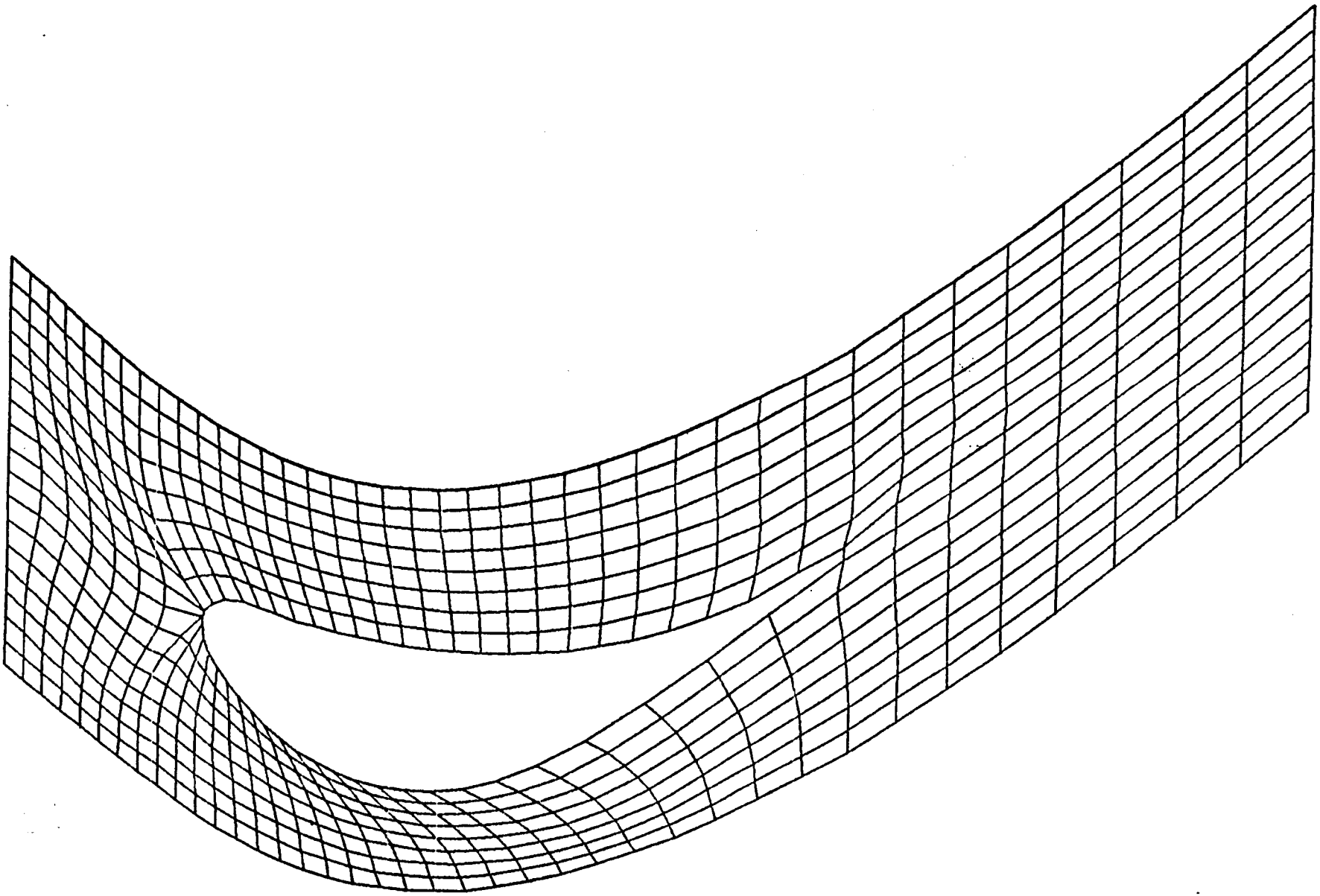

Fig. 5 Rotor Grid - Coarse Mesh (40 x 18)

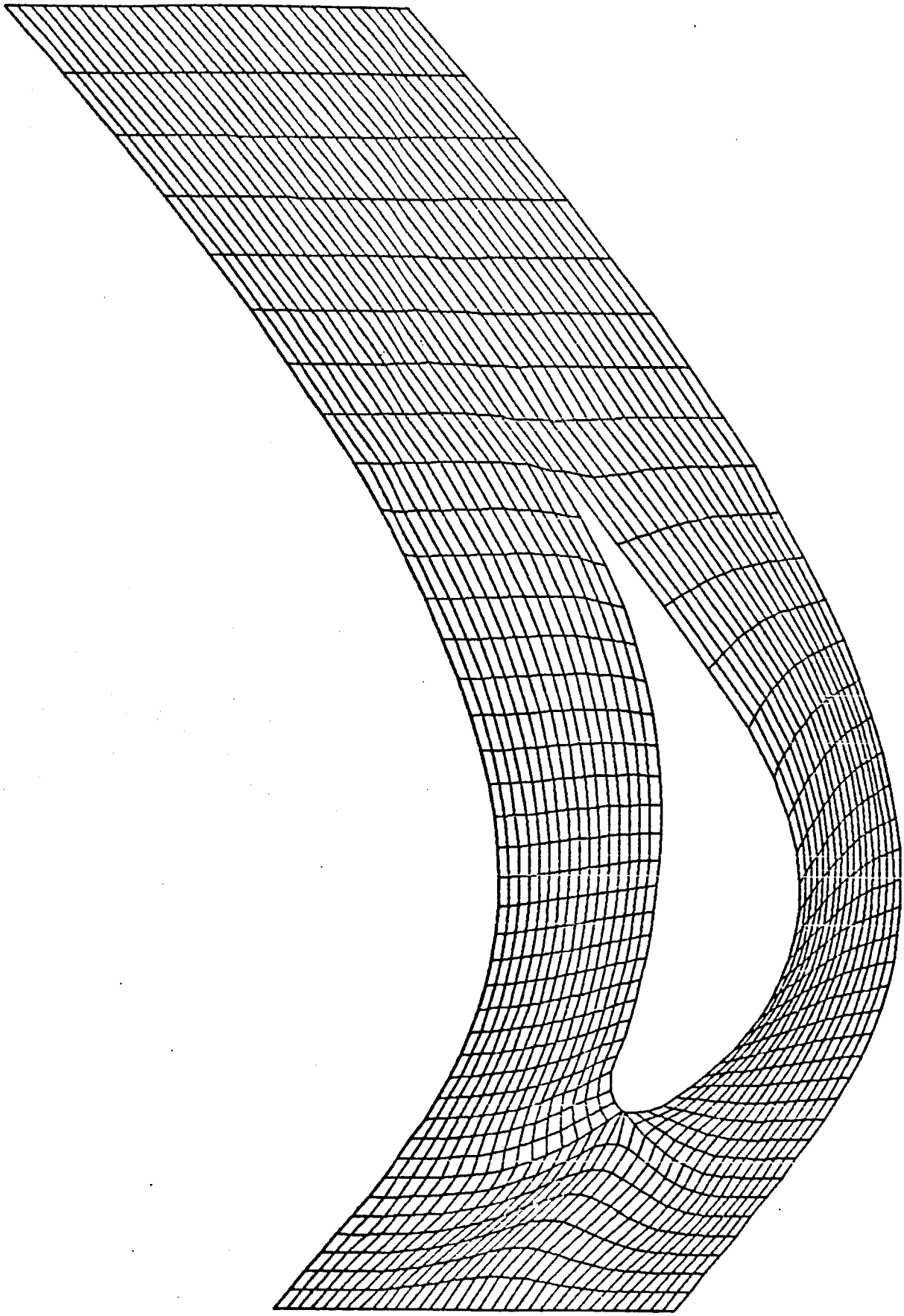

Fig. 6 Rotor Grid - Fine Mesh (40 x 34)

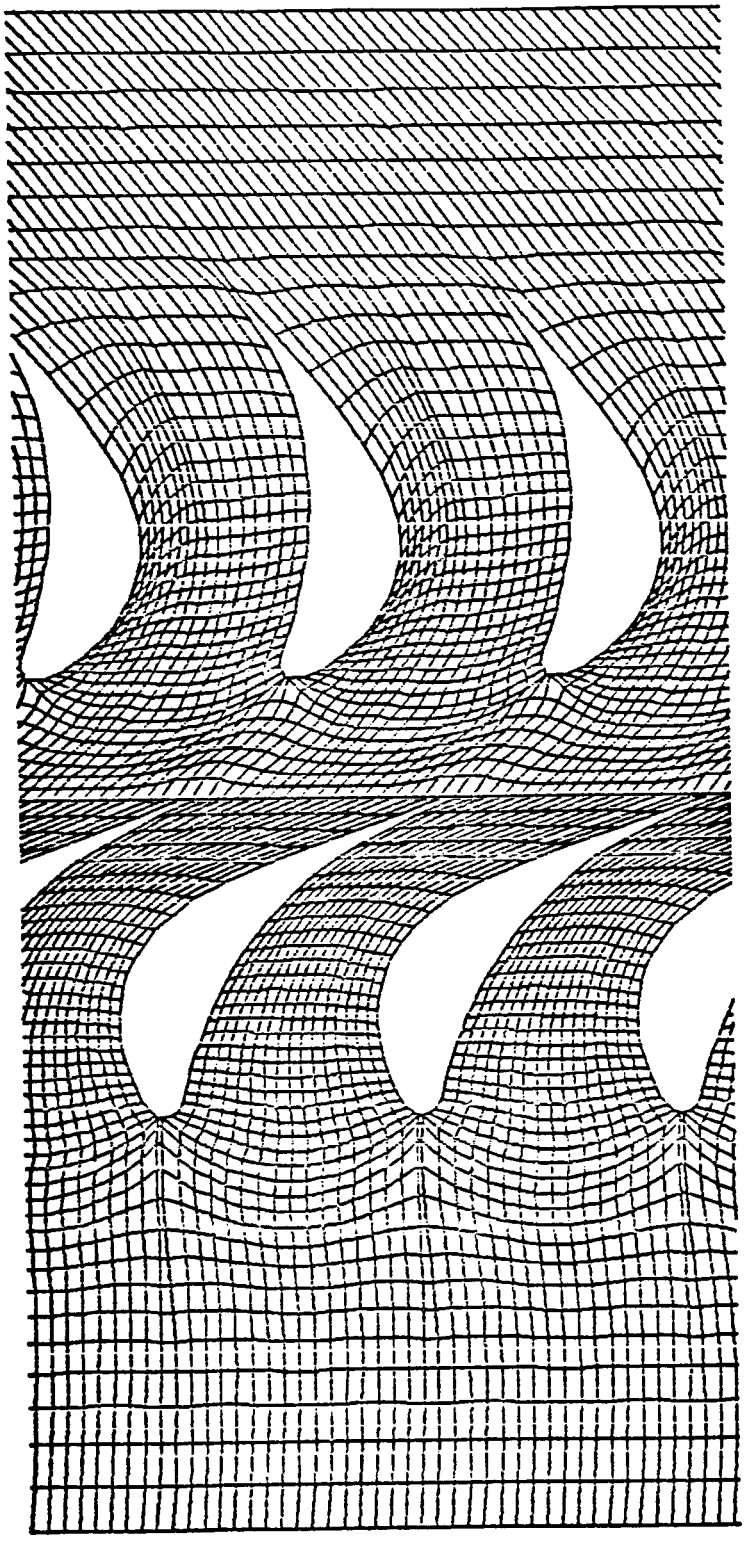

Fig. 7 Composite Stator - Rotor Grid (80 x 18)

```
      PARAMETER (kx=189,LY=34)
CCC   IMPLICIT REAL*8 (A-H,O-Z)
      COMMON      XU(KX,LY),XL(KX,LY),YU(KX,LY),YL(KX,LY),
     1           SCAL,NX,NY,ILE,ITE
      COMMON/MSH/ A0(KX),B0(67),XSING,YSING,XLIM,BOUND,HWALL
      COMMON/GEO/ XP(KX),YP(KX),TRAIL,SLOPT,NP
      COMMON/FLO/ FMACH,ALPHA,CA,SA,CIRC,KSYM
      COMMON/RLX/ FRES,ARES,DG,AG,P1,P2,P3,P4,P5,VIS,QC,
     1           IRES,JRES,IG,JG,NSUP,KODE,MODE
      COMMON/OUT/ SVU(65),SVL(65),SMU(65),SML(65),CPU(65),CPL(65)
     DIMENSION TOT0(6),COV0(6),P10(6),P20(6),P30(6),P40(6),P50(6),
     2 GMESH(6),TITLE(20),RES(502),COUNT(502)
      IREAD     = 5
      IWRIT     = 6
      RAD       = 57.295779513082
      READ  (IREAD,530) (TITLE(I),I=1,20)
      WRITE (IWRIT,630) (TITLE(I),I=1,20)
C     READS PROFILE DATA FOR CAMBER REMOVED CASCADE
      READ  (IREAD,500)
      READ  (IREAD,510) FSYM,FNU,FNL,FNX,FNY,FMESH,VIS,QC
      ISYM      = FSYM
      NU        = FNU
      NL        = FNL
C     IF (NU.LT.1) GO TO 301
      NX        = FNX
      NY        = FNY
      MMESH     = FMESH
      IF (QC.LE.0.) QC = .9
      READ  (IREAD,500)
      DO 4 M=1,MMESH
      READ  (IREAD,510) TOT0(M),COV0(M),P10(M),P20(M),P30(M),
     1                  P40(M),P50(M),GMESH(M)
    4 CONTINUE
      READ (IREAD,500)
      READ (IREAD,510) FMIT1,FMIT2,HWALL
      MIT1      = FMIT1
      MIT2      = FMIT2
      READ  (IREAD,500)
      READ  (IREAD,510) AL1,AL2,STEP,FM1,FM2,DMACH
      IF (STEP.LE.0.) AL2 = AL1  -STEP  -1.
      IF (DMACH.LE.0.) FM2 = FM1  -DMACH  -1.
      AL        = AL1
      FMACH     = FM1
      READ  (IREAD,500)
      READ  (IREAD,510) TRAIL,SLOPT,XSING,YSING
      NP        = NL  +NU  -1
      READ  (IREAD,500)
      DO 12 I=NL,NP
      READ  (IREAD,510) XP(I),YP(I)
      XP(I)  = XP(I)*2.0
   12  CONTINUE
      L         = NL  +1
      IF (ISYM.GT.0) GO TO 15
      READ  (IREAD,500)
```

```
      DO 14 I=1,NL
      READ  (IREAD,510) VAL,DUM
      J          = L  -I
      XP(J)      = VAL
   14 YP(J)      = DUM
      GO TO 21
   15 J          = L
      DO 16 I=NL,NP
      J          = J  -1
      XP(J)      = XP(I)
   16 YP(J)      = -YP(I)
   21 CHORD      = XP(1)  -XP(NL)
       ALPHA1=0./RAD
      L1=NL+1
      J=L1
       ISYM=0
      XM         = XP(NL)  +.25*CHORD
C     WRITE (IWRIT,32)
C  32 FORMAT(18HOINPUT COORDINATES/
C     1       15HO       X      ,15H        Y      )
C     DO 34 I=1,NP
C  34 WRITE (IWRIT,610) XP(I),YP(I)
C     WRITE (IWRIT,36)
C  36 FORMAT(15HO     XSING    ,15H      YSING    ,15H     TE SLOPE  ,
C     1       15H     TE ANGLE  )
C     WRITE (IWRIT,610) XSING,YSING,SLOPT,TRAIL
      TRAIL      = TRAIL/RAD
      IF (MMESH.EQ.1) GO TO 51
      DO 42 M=2,MMESH
      NX         = NX  +NX
   42 NY         = NY  +NY
   51 CALL COORD
      CALL GEOM
      CALL MESH
         WRITE(6,1220) SCAL,ITE,ILE
          WRITE(6,*) CHORD
 1220   FORMAT(E15.8,2I3)
c     WRITE (IWRIT,600)
c     WRITE (IWRIT,62)
      DO 64 I=1,MX
      XA         = SCAL*XU(I,1)
      YA         = SCAL*YU(I,1)
      XB         = SCAL*XU(I,JU)
      YB         = SCAL*YU(I,JU)
   64 WRITE (IWRIT,680) I,A0(I),XA,YA,XB,YB
c     WRITE (IWRIT,600)
c     WRITE (IWRIT,66)
   66 FORMAT(5H    J,15H          BO   ,
     1              15H         X(1,J),
     2              15H         Y(1,J),15H        X(MX,J),
     3              15H        Y(MX,J))
      KY         = NY  +2
         STOP
  500 FORMAT(1X)
```

```
  510 FORMAT(8E10.6)
  530 FORMAT(20A4)
  600 FORMAT(1H1)
  610 FORMAT(E12.4,8E15.4)
  630 FORMAT(1H0,20A4)
  640 FORMAT(I8,7I15)
  650 FORMAT(I7,E12.4,2I5,2E12.4,2I5,E12.4,2F10.4,I7)
  660 FORMAT(E12.4,E15.4,3E15.4)
  670 FORMAT(F10.6,8I6)
  680 FORMAT (I5,7F15.4)
  700 FORMAT(15HOCOMPUTING TIME,F10.3,10H   SECONDS)
          END
       SUBROUTINE  COORD
       PARAMETER (kx=189,LY=34)
CCC    IMPLICIT REAL*8 (A-H,O-Z)
       COMMON      XU(KX,LY),XL(KX,LY),YU(KX,LY),YL(KX,LY),
     1            SCAL,NX,NY,ILE,ITE
       COMMON/MSH/ AO(KX),BO(67),XSING,YSING,XLIM,BOUND,HWALL
C      SET UP THE GRID BOUNDARIES AND DETERMINE NUMBER OF POINTS
C      UPSTREAM ON THE BODY AND DOWNSTREAM
       AY          = 1.
       AX          = 1.
       XLIM        = 0.5
       BOUND       = 0.95
       XMAX        = XLIM*BOUND
       SY          = .5
       YMAX        = HWALL*BOUND
       S           = 1.
CCC      S           = 0.5
       LX          = NX/2  +1
       MX          = NX  +1
       DX          = 2.*BOUND/NX
       L           = 1.000001*XMAX/DX
       ILE         = LX  -L
       ITE         = LX  +L
       DO 12 I=1,MX
       D           = (I  -LX)*DX
C      FOR NO STRETCHING COMMENT THE NEXT SIX LINES
CC       IF (ABS(D).LE.XMAX) GO TO 12
CCC     B          = 1.
CCC    IF (D.LT.0.) B = -1.
CCC    A           = 1.  -((D  -B*XMAX)/(1.  -XMAX))**2
CCC    C            = A**AX
CCC    D           = B*XMAX  +(D  -B*XMAX)/C
  12   AO(I)      = D
       JU          = NY/2  +1
       MY          = NY  +2
       DY          = 2./NY
       DO 22 J=1,JU
       D           = (JU  -J)*DY  -0.5
       SGN         = SIGN (1.,D)
C      FOR NO STRETCHING S=1.0 (J-DIRECTION)
C  22 BO(J) = YMAX*.5*(SGN*(2.*SGN*D)**S  +1.)
   22 BO(J) = YMAX*.5*(SGN*(2.*SGN*D)**S  +1.)
```

```
      DO 24 J=1,JU
      J1        = JU  +J
      J2        = JU  +1  -J
   24 B0(J1) = -B0(J2)
      RETURN
      END
C
C
C
      SUBROUTINE  GEOM
      PARAMETER (kx=189,LY=34)
CCC   IMPLICIT REAL*8 (A-H,O-Z)
      COMMON      XU(KX,LY),XL(KX,LY),YU(KX,LY),YL(KX,LY),
     1           SCAL,NX,NY,ILE,ITE
      COMMON/MSH/ A0(KX),B0(67),XSING,YSING,XLIM,BOUND,HWALL
      COMMON/GEO/ XP(KX),YP(KX),TRAIL,SLOPT,NP
      COMMON/MP1/ XS(KX),YS(KX),D1(KX),D2(KX),D3(KX)
      PI        = 3.1415926535898
      MX        = NX  +1
      SCAL      = 2.000002*XLIM*BOUND/(XP(NP)  -XSING)
      ANGL      = ATAN(SLOPT)
      ANGL1     = ATAN2((YP(1)  -YSING),(XP(1)  -XSING))
      ANGL2     = ATAN2((YP(NP)  -YSING),(XP(NP)  -XSING))
      ANGL1     = ANGL  -.5*(ANGL1  -TRAIL)
      ANGL2     = ANGL  -.5*(ANGL2  +TRAIL)
      T1        = TAN(ANGL1)
      T2        = TAN(ANGL2)
      ANGL      = PI  +PI
      U         = 1.
      V         = 0.
      DO 12 I=1,NP
      XA        = XP(I)  -XSING
      YA        = YP(I)  -YSING
      ANGL      = ANGL  +ATAN2((U*YA  -V*XA),(U*XA  +V*YA))
      R         = SCAL*SQRT(XA**2  +YA**2)
      U         = XA
      V         = YA
      R         = SQRT(R)
      XS(I)     = R*COS(.5*ANGL)
   12 YS(I)     = R*SIN(.5*ANGL)
      SCAL      = 1./SCAL
      CALL SPLIF (1,NP,XS,YS,D1,D2,D3,1,T1,1,T2,0,0.,IND)
      RETURN
      END

C
C
      SUBROUTINE  MESH
      PARAMETER (kx=189,LY=34)
CCC   IMPLICIT REAL*8 (A-H,O-Z)
C     GENERATES MESH
      COMMON      XU(KX,LY),XL(KX,LY),YU(KX,LY),YL(KX,LY),
     1           SCAL,NX,NY,ILE,ITE
      COMMON/MSH/ A0(KX),B0(67),XSING,YSING,XLIM,BOUND,HWALL
```

```
      COMMON/GEO/ XP(KX),YP(KX),TRAIL,SLOPT,NP
      COMMON/MP1/ XS(KX),YS(KX),D1(KX),D2(KX),D3(KX)
      DIMENSION   P(KX),Q(KX),S(KX)
      PI          = 3.1415926535898
      MX          = NX  +1
      MY          = NY  +2
      JU          = NY/2  +1
      X0          = XSING/SCAL
      Y0          = YSING/SCAL
 1221 FORMAT(4F10.6)
      XTE         = SQRT(XLIM*BOUND  +XLIM*BOUND)
      YTE         = YS(NP)
      XFF         = SQRT(A0(MX)  +XLIM*BOUND)
      A1          = .1
         WRITE(20,1012) MX,MY
 1012 FORMAT(2I8)
 1013 FORMAT(2F12.6)
      DO 16 J=1,MY
      I1          = 0
      DO 10 I=1,MX
      A           = A0(I)  +XLIM*BOUND
      R           = SQRT(A*A  +B0(J)*B0(J))
C     TYPE*,J,I,R,XTE,A1
C     COMPRESS THE GRID NEAR L.E
      IF(ABS(R).LE.0.00001) GO TO 1014
      R           = ((XTE  +A1)/XTE)*R/(SQRT(R)  +A1)
      GO TO 1015
 1014 R=0.0
 1015 CONTINUE
      ANGL        = 0.
      IF (R.GT.0.) ANGL = .5*ATAN2(B0(J),A)
      IF (J.LE.JU.AND.ANGL.LT.-.25*PI) ANGL = ANGL  +PI
      IF (J.GT.JU.AND.ANGL.LT..25*PI) ANGL = ANGL  +PI
      P(I)        = R*COS(ANGL)
      Q(I)        = R*SIN(ANGL)
      IF (ABS(P(I)).LE.XTE.AND.I1.EQ.0) I1 = I
      IF (ABS(P(I)).LE.XTE) I2 = I
   10 S(I)        = 0.
      IF (I1.NE.0) CALL INTPL (I1,I2,P,S,1,NP,XS,YS,D1,D2,D3,0)
      DO 16 I=1,MX
      Q(I)        = Q(I)  +S(I)
      IF (J.GE.JU+1) GO TO 14
      XU(I,J)     = X0  +P(I)*P(I)  -Q(I)*Q(I)
      YU(I,J)     = Y0  +2.*P(I)*Q(I)
      GO TO 16
   14 JJ          = MY  +1  -J
      XL(I,JJ)    = X0  +P(I)*P(I)  -Q(I)*Q(I)
      YL(I,JJ)    = Y0  +2.*P(I)*Q(I)
   16 CONTINUE
      MY2=MY/2
C     OUTPUT GRID COORDINATES FOR USE IN GRID2
      DO 551 J=1,MY2
      DO 551 I=1,MX-18
      WRITE(20,1013) XU(I,J),YU(I,J)
```

```
  551  CONTINUE
       DO 550 J=1,MY2
       DO 550 I=1,MX-18
       WRITE(20,1013) XL(I,J),YL(I,J)
  550  CONTINUE
       RETURN
       END
       SUBROUTINE  SPLIF(M,N,S,F,FP,FPP,FPPP,KM,VM,KN,VN,MODE,FQM,IND)
CCC    IMPLICIT REAL*8 (A-H,O-Z)
C      SPLINE FIT
C      INTEGRAL PLACED IN FPPP IF MODE GREATER THAN 0
C      IND SET TO ZERO IF DATA ILLEGAL
       DIMENSION   S(1),F(1),FP(1),FPP(1),FPPP(1)
       IND        = 0
       K          = IABS(N  -M)
       IF (K  -1) 81,81,1
    1  K          = (N  -M)/K
       I          = M
       J          = M  +K
       DS         = S(J)  -S(I)
       D          = DS
       IF (DS) 11,81,11
   11  DF         = (F(J)  -F(I))/DS
       IF (KM  -2) 12,13,14
   12  U          = .5
       V          = 3.*(DF  -VM)/DS
       GO TO 25
   13  U          = 0.
       V          = VM
       GO TO 25
   14  U          = -1.
       V          = -DS*VM
       GO TO 25
   21  I          = J
       J          = J  +K
       DS         = S(J)  -S(I)
       IF (D*DS) 81,81,23
   23  DF         = (F(J)  -F(I))/DS
       B          = 1./(DS  +DS  +U)
       U          = B*DS
       V          = B*(6.*DF  -V)
   25  FP(I)      = U
       FPP(I)     = V
       U          = (2.  -U)*DS
       V          = 6.*DF  +DS*V
       IF (J  -N) 21,31,21
   31  IF (KN  -2) 32,33,34
   32  V          = (6.*VN  -V)/U
       GO TO 35
   33  V          = VN
       GO TO 35
   34  V          = (DS*VN  +FPP(I))/(1.  +FP(I))
   35  B          = V
       D          = DS
```

```
   41 DS         = S(J)  -S(I)
      U          = FPP(I)  -FP(I)*V
      FPPP(I)    = (V  -U)/DS
      FPP(I)     = U
      FP(I)      = (F(J)  -F(I))/DS  -DS*(V  +U  +U)/6.
      V          = U
      J          = I
      I          = I  -K
      IF (J  -M) 41,51,41
   51 I          = N  -K
      FPPP(N)    = FPPP(I)
      FPP(N)     = B
      FP(N)      = DF  +D*(FPP(I)  +B  +B)/6.
      IND        = 1
      IF (MODE) 81,81,61
   61 FPPP(J)    = FQM
      V          = FPP(J)
   71 I          = J
      J          = J  +K
      DS         = S(J)  -S(I)
      U          = FPP(J)
      FPPP(J)    = FPPP(I)  +.5*DS*(F(I)  +F(J)  -DS*DS*(U  +V)/12.)
      V          = U
      IF (J  -N)  71,81,71
   81 RETURN
      END
C
C
C
      SUBROUTINE  INTPL(MI,NI,SI,FI,M,N,S,F,FP,FPP,FPPP,MODE)
CCC   IMPLICIT REAL*8 (A-H,O-Z)
C     INTERPOLATION USING TAYLOR SERIES
C     ADDS CORRECTION FOR PIECEWISE CONSTANT FOURTH DERIVIATIVE
C     IF MODE GREATER THAN 0
      DIMENSION   SI(1),FI(1),S(1),F(1),FP(1),FPP(1),FPPP(1)
      K          = IABS(N  -M)
      K          = (N  -M)/K
      I          = M
      MIN        = MI
      NIN        = NI
      D          = S(N)  -S(M)
      IF (D*(SI(NI)  -SI(MI))) 11,13,13
   11 MIN        = NI
      NIN        = MI
   13 KI         = IABS(NIN  -MIN)
      IF (KI) 21,21,15
   15 KI         = (NIN  -MIN)/KI
   21 II         = MIN  -KI
      C          = 0.
      IF (MODE) 31,31,23
   23 C          = 1.
   31 II         = II  +KI
      SS         = SI(II)
   33 I          = I  +K
```

```
      IF (I  -N) 35,37,35
   35 IF (D*(S(I)  -SS)) 33,33,37
   37 J          = I
      I          = I  -K
      SS         = SS  -S(I)
      FPPPP      = C*(FPPP(J)  -FPPP(I))/(S(J)  -S(I))
      FF         = FPPP(I)  +.25*SS*FPPPP
      FF         = FPP(I)  +SS*FF/3.
      FF         = FP(I)  +.5*SS*FF
      FI(II)     = F(I)  +SS*FF
      IF (II  -NIN) 31,41,31
   41 RETURN
      END
C
C
C
```

TURBINE GRID (PROFILE)

FSYM	FNU	FNL	FNX	FNY	FMESH	VIS	QC
0.0	19.	19.	52.	16.	1.	1.0	0.90

TOTO	COVO	P10	P20	P30	P40	P50	GMESH
32.	0.	0.000	0.00	0.0	0.00	0.00	1.0

FMIT1	FMIT2	HWAL
1.0	1.0	0.220

AL1	AL2	STEP	FM1	FM2	DMACH
0.000	0.	0.	.500	0.	0.

TRAIL	SLOPT	XSING	YSING
5.0	0.0	.001050	0.000

XP(I)	YP(I)
0.000000	0.000000
0.015700	0.027706
0.108800	0.088597
0.212100	0.137853
0.315400	0.180847
0.399900	0.212482
0.496500	0.242702
0.568900	0.259087
0.629300	0.266360
0.689700	0.264002
0.738000	0.250662
0.786200	0.222683
0.822100	0.192298
0.849100	0.167692
0.885000	0.133698
0.911800	0.108015
0.944000	0.079050
0.976200	0.053698
0.999000	-0.002591

XP(I)	YP(I)
0.000000	0.000000
0.009900	-0.044558
0.037800	-0.059870
0.123200	-0.072826
0.208700	-0.082956
0.294100	-0.089020
0.379500	-0.090128
0.447900	-0.086748
0.533300	-0.077099
0.601600	-0.064913
0.670000	-0.049004
0.738300	-0.030830
0.789500	-0.016857
0.840800	-0.004127
0.875000	0.001918
0.926200	0.003198
0.960300	-0.006010
0.974700	-0.012163
0.999000	-0.002591

PR1	PR2	BUNCH	IRD	IWRT	IGSR (3F10.6,3I6)
1.	1.0	.0	-2	-1	+0

```
      PARAMETER(KX =130,KY=67)
C     PROGRAM GRID2 (FOR CASCADE GEOMETRIES-USES OUTPUT FROM GRID1
      DIMENSION XVM(KX,KY),YVM(KX,KY),DXP(KX),DYP(KX),C(KY),EP(KY),EM(KY)
      DIMENSION XVP(KX,KY),YVP(KX,KY),XN(KX,KY),YN(KX,KY)
      DIMENSION XE(KX,KY),YE(KX,KY),EI(KX),EJC(KX),EIC(KX)
      DIMENSION XU(KX),YCU(KX),YCL(KX)
      DIMENSION XI(10),YI(10),XII(5),YII(5),E1(10),E2(10),E3(10)
      DIMENSION YVPN(KX,KY),YNN(KX,KY),E1(KX)
      DIMENSION XEE(KX,KY),YEE(KX,KY)
      DIMENSION YO1(KX),YO2(KX),YU(KX),YL(KX)
      PI=4.*ATAN(1.0)
      IWRIT = 1
      DELTG = 0.0461
      GSCAL = 2.53
      DTR = 180./PI
      THETA = 32.25/DTR
      ALPHA=-2.5/DTR
      A1    =  1.0113
      A2    =  0.0000
      A3    = -0.4240
      A4    = -0.4484
C     THE NEXT FOUR LINES DEFINE LOCATION OF THE CASCADE
      JBODY = 9
      JBP    = JBODY +1
      ILE   = 15
      ITE =   39
      IR  = 1
      NPLT= 1
      STR = 0.5
C     INPUT PROGRAM INITIALISATION PARAMETERS
C     IREV =1 STATOR GRID, 0 -ROTOR GRID
C     NAV  = NO. OF TIMES SMOOTHING APPLIED (ABOUT 25 TO 50)
C     ICR  = 0 COARSE GRID 1 FINE GRID (UPSTREAM)
C     INPUT FROM UNIT 8 THE TWO DIMENSION GRID GENERATED BY GRID1.FOR
      READ(5,*) IREV,NAV,ICR
      READ(8,*) NX,NY
      DO 20 J=1,NY
      DO 20 I=1,NX
      READ(8,*)XVP(I,J),YVP(I,J)
      XVP(I,J)=XVP(I,J)/GSCAL
      YVP(I,J)=YVP(I,J)/GSCAL
   20 CONTINUE
C     GENERATE THE CARTESIAN GRID WITH CORRECT OUTER PERIODIC SHAPE
C     THIS IS USED TO BLEND WITH THE GRID INPUT EARLIER
      XBG =XVP(1,1)
      YBG =YVP(1,1)
      DELTX =ABS(XVP(1,1)-XVP(2,1))*.93
      DO 25 I=1,NX
      YN(I,1) = YBG
      YNN(I,1) = YBG
      XN(I,1) = XBG +DELTX*(I-1)
   25 CONTINUE
      DO 30 J=2,NY
      DELTY =-ABS(YVP(1,1)-YVP(1,2))
```

```
      JJ =J
      C1 = 1.38
      C1R= 1.1
      IF (J.GE.JBP) THEN
      C1 = .82
      C1R =1.1
      END IF
      IF (J.EQ.JBP) DELTY = 0.00000000
      IF (J.GE.JBP) JJ=J-1
      DO 30 I=1,NX
      XN(I,J)=XBG+DELTX*(I-1)
      YN(I,J)=YN(I,J-1) +DELTY*0.773*C1
      YNN(I,J)=YNN(I,J-1) +DELTY*0.773*C1R
 30   CONTINUE
C     DEFINE  BLENDING FUNCTIONS IN I AND J DIRECTIONS
      ITE1C=ILE
      ITE1S= ILE/2 +1
      ITE2C=ITE
      DO 35 I=1,ITE1C
      FIC(I)=0.5*(1.-COS(FLOAT(I-1)*PI/FLOAT(ITE1C-1)))
      F1(I) =0.5*(1.-COS(FLOAT(I-ITE1S)*PI/FLOAT(ITE1C-ITE1S)))
      IF (I.LE.ITE1S) F1(I) =0.0
 35   CONTINUE
      DO 95 J=1,NYC2
      FJC(J)=0.5*(1.-COS(FLOAT(J-1)*PI/FLOAT(NYC2-1)))
 95   CONTINUE
      DO 98 J=NYC2+1,NYC
      J1=NYC+1-J
      FJC(J)=0.5*(1.-COS(FLOAT(J1-1)*PI/FLOAT(NYC-NYC2-1)))
C     FJC(J)=1.
98    CONTINUE
      ITE2M = ITE2C
      DO 45 I=ITE1C,ITE2M
      F1(I) = 1.
      FIC(I)=1.
 45   CONTINUE
      ITP = ITE+1
      DO 80 I=ITE2M+1,NX
      I1=NX-I+1
      FIC(I)=0.5*(1.-COS(FLOAT(I1-1)*PI/FLOAT(NX-ITE2M)))
      F1(I) = FIC(I)
 80   CONTINUE
C     THE SPLITTING OF GRID LINE AHEAD OF LEADING EDGE AND
C     BEYOND TRAILING EDGE
      FACT  = 0.5
      IF (ICR.EQ.1) FACT =.25
      DELTYG  = -DELTG*FACT
      ITE2M   = ITE2C-1
      IFF     = ITP
      XILE = XVP(ITE1C,JBODY)
      YILE = YVP(ITE1C,JBODY)
      DO 85 I=ITE2M,IFF
      I1=ITP-I+1
      FI(I)=0.5*(1.-COS(FLOAT(I1-1)*PI/FLOAT(ITP-ITE2M)))
```

```
 85     CONTINUE
        DO 88  J=1,NY
        IFACT  = -1
        IF (J.GE.JBP) IFACT = 1
        XVP(ITE1C,J) = 0.5*(XVP(ITE1C,J)+XVP(ITE1C+1,J))
        DO 11 I=1,ITE1C
        FACT   = FIC(I)*0.75
        IF (IREV.EQ.0) FACT =FI(I)*0.75
        YVP(I,J) = YVP(I,J)+DELTYG*IFACT*(1.-FACT)
        YNN(I,J) = YNN(I,J)+DELTYG*IFACT*(1.-FACT)
        YN(I,J)  = YN(I,J)+DELTYG*IFACT*(1.-FACT)
 11     CONTINUE
        IEND = ITP
        IF (IREV.EQ.0) IEND =NX
        DO 12 I=ITE2C-1,IEND
        FACT     = FI(I)
        YVP(I,J) = YVP(I,J)+DELTYG*IFACT*(1.-FACT*0.75)
        YNN(I,J) = YNN(I,J)+DELTYG*IFACT*(1.-FI(I)*0.75)
 12     YN(I,J)  = YN(I,J)+DELTYG*IFACT*(1.-FI(I)*0.75)
88     CONTINUE
C      INTERPOLATE FOR BODY POINTS ON EITHER SIDE OF L.E AFTER
C      SPLITTING
       XI(1) = XVP(ITE1C+2,JBODY)
       YI(1) = YVP(ITE1C+2,JBODY)
       XI(2) = XVP(ITE1C+1,JBODY)
       YI(2) = YVP(ITE1C+1,JBODY)
       XI(3) = XILE
       YI(3) = YILE
       XI(4) = XVP(ITE1C+1,JBP)
       YI(4) = YVP(ITE1C+1,JBP)
       XI(5) = XVP(ITE1C+2,JBP)
       YI(5) = YVP(ITE1C+2,JBP)
       YII(1)= YVP(ITE1C,JBODY)
       YII(2)= YVP(ITE1C,JBP)
       MI =1
       CALL SPLIF(1,5,YI,XI,E1,E2,E3,2,0.,2,0.,0,0.,IND)
       CALL INTPL(MI,2,YII,XII,1,5,YI,XI,E1,E2,E3,0)
       XVP(ITE1C,JBODY) =XII(1)
       XVP(ITE1C,JBP)=XII(2)
 143   CONTINUE
       DO 90 J=1,NY
       DO 90 I=1,NX
       YNN(I,J)=YNN(I,J)+DELTY*1.98
 90    CONTINUE
       NYC2=NY/2
       NYC=NY
C      GRID BLENDING
       DO 100 J =1,NY
       DO 100 I =1,NX
       FW = FJC(J)*FJC(J)*FIC(I)*FIC(I)
       FW = FW*FW
       XN(I,J)=(1.-FW)*XN(I,J)+FW*XVP(I,J)
       YN(I,J)=(1.-FW)*YN(I,J)+FW*YVP(I,J)
       YNN(I,J)=(1.-FW)*YNN(I,J)+FW*YVP(I,J)
```

```
 100  CONTINUE
      DO 105 J = 1,NY
      IF (J.GT.JBODY) THEN
      JM = NY
      JB =JBP
      ELSE
      JM =1
      JB =JBODY
      END IF
      VAL = 0.4
      DO 105 I=1,NX
      CON = (YN(I,JM)*VAL-YN(I,JB))/(YN(I,JM)-YN(I,JB))
      YVP(I,J)=YN(I,JB)+CON*(YN(I,J)-YN(I,JB))
      CON = (YNN(I,JM)*VAL-YNN(I,JB))/(YNN(I,JM)-YNN(I,JB))
      YVPN(I,J)=YNN(I,JB)+CON*(YNN(I,J)-YNN(I,JB))
 105  CONTINUE
      DO 110 J=1,NY
      DO 110 I=1,NX
      YN(I,J)=YVP(I,J)
      YNN(I,J)=YVPN(I,J)
 110  CONTINUE
      IF (IREV.GE.5) GO TO 2223
C     ADD CAMBER LINE DESCRIBED BY Y = A1+A2*X+A3*X*X+A4*X*X*X
      DO 115 J=1,NY
      DO 115 I=1,NX
      XU(I)=XN(I,J)
      YCU(I)=A1-A3*XU(I)*XU(I)-A4*XU(I)*XU(I)*XU(I)
      YVP(I,J)=YNN(I,J)+YCU(I)
      YN(I,J)=YN(I,J)+YCU(I)
      YVP(I,J)=-YVP(I,J)
      XVP(I,J)=XN(I,J)
      XVM(I,J)=XVP(I,J)
      YVM(I,J)=YVP(I,J)
 115  CONTINUE
      DO 120 J=1,NY
      JJ = NY+1-J
      DO 120  I=1,NX
      XVP(I,JJ)=XVM(I,J)
      YVP(I,JJ)=YVM(I,J)
120   CONTINUE
C     ROTATE THE ROTOR GRID BY THETA ABOUT L.E
      X11  = (XVP(ITE1S,JBODY)+XVP(ITE1S,JBP))*0.5
      Y11  = (YVP(ITE1S,JBODY)+YVP(ITE1S,JBP))*0.5
      DO 125 J=1,NY
      JB  = JBODY
      IF (J.GE.JBP)JB =JBP
      DO 125 I=1,NX
      X1  =XVP(ITE1S,J)
      Y1  =YVP(ITE1S,J)
      IF (J.NE.JB) GO TO 222
      IF (I.GE.ITE1C.AND.I.LE.ITE2C) THEN
      X1  = X11
      Y1  = Y11
      END IF
```

```
222   CONTINUE
      XV =XVP(I,J)
      XVP(I,J)=X1+(XVP(I,J)-X1)*COS(THETA)+(YVP(I,J)-Y1)*SIN(THETA)
      YVP(I,J)=Y1+(YVP(I,J)-Y1)*COS(THETA)-(XV-X1)*SIN(THETA)
125   CONTINUE
C     GRID SMOOTHING
      DO 128 IAV =1,NAV
      DO 121 J=1,NY
      DO 121 I=1,NX
      XVM(I,J)=XN(I,J)
      YVM(I,J)=YN(I,J)
121   CONTINUE
      JBMM = JBODY -1
      JBPP = JBP+1
      JNORM= NY -JBODY
      DO 122 J=2,NY-1
      IF (J.LE.JBODY) THEN
      ALPHA =1.-FLOAT(JBMM-J)/FLOAT(JNORM)
      ELSE
      ALPHA =1.- FLOAT(J-JBPP)/FLOAT(JNORM)
      END IF
      IF (J.EQ.JBODY.OR.J.EQ.JBP) GO TO 122
      DO 122 I=2,ITP
      XN(I,J)=(XVM(I+1,J)+XVM(I-1,J)+
     1ALPHA*XVM(I,J+1)+XVM(I,J-1)*ALPHA)/(2.+2.*ALPHA)
      YN(I,J)=(YVM(I,J+1)+YVM(I,J-1))*.5
122   CONTINUE
126   CONTINUE
      ILE1 =ILE -1
      JBO = JBODY
      XAV = 0.5*(XN(ILE1,JBO)+XN(ILE1,JBO+1))
      YAV = 0.5*(YN(ILE1,JBO)+YN(ILE1,JBO+1))
        XN(ILE1+1,JBO+1) = 0.25*(XN(ILE1+1,JBO+1)*2.+XN(ILE1,JBO+1)
     1+XN(ILE1+2,JBO+1))
        YN(ILE1+1,JBO+1) = 0.25*(YN(ILE1+1,JBO+1)*2.+YN(ILE1,JBO+1)
     1+YN(ILE1+2,JBO+1))
        XAV = 0.5*(XVP(ILE1,JBO)+XVP(ILE1,JBO+1))
        YAV = 0.5*(YVP(ILE1,JBO)+YVP(ILE1,JBO+1))
        XVP(ILE1+1,JBO) = 0.25*(XVP(ILE1+1,JBO)*2.+XVP(ILE1,JBO)
     1+XVP(ILE1+2,JBO))
        YVP(ILE1+1,JBO) = 0.25*(YVP(ILE1+1,JBO)*2.+YVP(ILE1,JBO)
     1+YVP(ILE1+2,JBO))
        DO 135 IAV =1,NAV
        DO 141 J=1,NY
        DO 141 I=1,NX
        XVM(I,J)=XVP(I,J)
        YVM(I,J)=YVP(I,J)
  141   CONTINUE
        DO 142 J=2,NY-1
        IF (J.LE.JBODY) THEN
        JBOD = JBODY
        ALPHA =1.-(FLOAT(JBMM-J)/JNORM)
        ELSE
        JBOD = JBP
```

```
      ALPHA =1.-(FLOAT(J-JBPP)/JNORM)
      END IF
      IF (J.EQ.JBODY.OR.J.EQ.JBP) GO TO 142
      DO 142 I=2,ITP
      XAV = (XVM(I+1,J)+XVM(I-1,J)+XVM(I,J+1)+XVM(I,J-1))*0.25
      YAV = (YVM(I+1,J)+YVM(I-1,J)+YVM(I,J+1)+YVM(I,J-1))*0.25
      IF (IAV.GT.10) ALPHA =0.0
      XVP(I,J)=ALPHA*XAV+(1.-ALPHA)*XVM(I,J)
      YVP(I,J)=(YVM(I,J+1)+YVM(I,J-1))*.5
 142  CONTINUE
 135  CONTINUE
C     STRETCHING OF STATOR GRID
      XTE =(XN(ITE,JBODY)+XN(ITE,JBP))*0.5
      YTE =(YN(ITE,JBODY)+YN(ITE,JBP))*0.5
      XN(ITE,JBODY) =XTE
      YN(ITE,JBODY) =YTE
      XN(ITE,JBP)=XTE
      YN(ITE,JBP)=YTE
      XTE =(XVP(ITE,JBODY)+XVP(ITE,JBP))*0.5
      YTE =(YVP(ITE,JBODY)+YVP(ITE,JBP))*0.5
      XVP(ITE,JBODY) =XTE
      YVP(ITE,JBODY) =YTE
      XVP(ITE,JBP)=XTE
      YVP(ITE,JBP)=YTE
      ILM = ILE-7
      DO 300 J=1,NY
      DO 300 I=1,ILE
      DELTI = FLOAT(ILE-I)/FLOAT(ILE-1)
      XN(I,J) = -STR*DELTI*DELTI*DELTI+XN(I,J)
 300  CONTINUE
      DO 16 J=1,NY
     SLOP=(YVP(ITE2C-2,J)-YVP(ITE2C-1,J))/
    1(XVP(ITE2C-2,J)-XVP(ITE2C-1,J))
      YVP(ITE2C,J)=YVP(ITE2C-1,J)+SLOP*(XVP(ITE2C,J)-XVP(ITE2C-1,J))
      SLOP=(YN(ITE2C-2,J)-YN(ITE2C-1,J))/(XN(ITE2C-2,J)-XN(ITE2C-1,J))
 16   YN(ITE2C,J)=YN(ITE2C-1,J)+SLOP*(XN(ITE2C,J)-XN(ITE2C-1,J))
C     SHIFT THE ROTOR GRID TO GET COMBINED GRID
      IF (IREV.EQ.1) GO TO 2223
      XVC = XVP(ILM,1)
      XC  = XN(ITP,1)
      XDIF  = XC - XVC
      IF (IREV.EQ.0) GO TO 2221
      DO  170 J=1,NY
      YC  = YN(ITP,J)
      YVC = YVP(ILM,J)
      DO  170 I = ILM,NX
      XVP(I,J) = XVP(I,J)+XC-XVC
 170  CONTINUE
      DO 175 J =1,NY
      XN(ITP,J)=XN(ITP,1)
      XC  = XN(ITP,J)
      YC  = YN(ITP,J)
      XVC = XVP(ILM,J)
      YVC = YVP(ILM,J)
```

```
      DO  175 I=ILM,NX
      YVP(I,J)=YVP(I,J)+(YC-YVC)+DY
  175 CONTINUE
      NXN   = ITP+1+NX-ILM
      DO 333 J=1,NY
      II=1
      DO 333 I=ITP+1,NXN
      IN =ILM+II
      XN(I,J)=XVP(IN,J)
      YN(I,J)=YVP(IN,J)+DY2
      II  =II+1
  333 CONTINUE
      NX =NXN-1
      GO TO 2223
 2221 CONTINUE
      DO 2222 J=1,NY
      DO 2222 I=1,NX
      I1   = I+7
      XN(I,J)=XVP(I1,J)+XDIF
      YN(I,J)=YVP(I1,J)
 2222 CONTINUE
 2223 CONTINUE
      IF (IREV.LE.1) THEN
      NX =ITP
      END IF
C     INTERPOLATE (LINEAR) TO FINE GRID FOR ICR = 1
      IF (ICR.NE.1) GO TO 610
      KYU = JBODY
      KYB = JBP
      JJ  = 0
      DO 500 J =1,KYU
      JJ  = 2*J-1
      JP  = JJ+1
      DO 500 I=1,NX
      XVP(I,JJ) = XN(I,J)
      YVP(I,JJ) = YN(I,J)
      IF (J.EQ.KYU) GO TO 500
      XVP(I,JP) = (XN(I,J)+XN(I,J+1))*0.5
      YVP(I,JP) = 0.5*(YN(I,J)+YN(I,J+1))
  500 CONTINUE
      DO 600 J =KYB,NY
      JJ   = 2*J-2
      JP   = JJ+1
      DO 600 I=1,NX
      XVP(I,JJ) = XN(I,J)
      YVP(I,JJ) = YN(I,J)
      IF (J.EQ.NY) GO TO 600
      XVP(I,JP) = (XN(I,J)+XN(I,J+1))*0.5
      YVP(I,JP) = 0.5*(YN(I,J)+YN(I,J+1))
  600 CONTINUE
      NY = 2*NY-2
      DO 610 J=1,NY
      DO 610 I=1,NX
      XN(I,J) = XVP(I,J)
```

```
      YN(I,J) = YVP(I,J)
 610  CONTINUE
      IF (IREV.EQ.1) GO TO 620
      DO 620 J=1,NY
      XN(NX,J) = XN(NX,1)
      XN(1,J)  = XN(1,1)
 620  CONTINUE
C     ADDITOINAL SMOOTHING FOR ROTOR GRID
      ISKIP = 1
      IF (IREV.LT.1) ISKIP = 0
      IF (ISKIP.NE.1) GO TO 1275
      JUP      = JBODY
      IF (ICR.GE.1) JUP =2*JBODY-1
      NXF      = NX
      IF(IREV.GT.1) NXF = ITP
      DO 1200 I=1,NXF
      YT       = YN(I,1)-YN(I,NY)
      YB1      = YN(I,JUP)
      YB2      = YN(I,JUP+1)
      YO1(I)   = YN(I,1)
      YO2(I)   = YN(I,NY)
      YU(I)    = (YT+YB1+YB2)/2.
1200  YL(I)    = (YB1+YB2-YT)/2.
      DO 1250 J=1,NY
      JBOD = JBODY
      IF (J.GE.JBP) JBOD = JBP
      IF (ICR.EQ.1) THEN
      JBOD     = 2*JBODY-1
      IF (J.GE.JBOD+1) JBOD = JBOD+1
      END IF
      DO 1250 I=1,NXF
      YBODY    = YN(I,JBOD)
      YMUL     = (YU(I)-YO1(I))/(YO1(I)-YBODY)
      IF (J.GT.JBODY+1) YMUL =(YL(I)-YO2(I))/(YO2(I)-YBODY)
      YN(I,J)= YN(I,J)+(YN(I,J)-YBODY)*YMUL
 1250 CONTINUE
 1275 CONTINUE
      JBO = JBODY
      IF (ICR.EQ.1) JBO = 2*JBODY -1
      KY2      = NY/2
      IF (IREV.NE.0) GO TO 881
      ILR =8
      DO 881 N =1,NAV-40
      DO 980 J=1,NY
      DO 980 I=1,NX
      XVM(I,J) =XN(I,J)
      YVM(I,J) =YN(I,J)
  980 CONTINUE
      DO 880 J=2,NY-1
      JP = 1
      JBOD = KY2
      IF (J.GT.KY2) JBOD = KY2+1
      IF (J.GT.KY2) JP =NY
      DO 880 I=2,ILR-1
```

```
        OMEGA = ABS(I-2)*ABS(J-JP)/(ABS(2.-ILR)*ABS(JP-JBOD))
        XAV   = 0.25*(XVM(I+1,J-1)+XVM(I+1,J+1)+XVM(I-1,J-1)+
     1XVM(I-1,J+1))
        YAV   = 0.25*(YVM(I,J-1)+YVM(I,J+1)+YVM(I,J+1)
     1+YVM(I,J-1))
        XN(I,J) = OMEGA*XAV+(1.-OMEGA)*XN(I,J)
        YN(I,J) =OMEGA*YAV+(1.-OMEGA)*YN(I,J)
 880    CONTINUE
 881    CONTINUE
C       INTERPOLATE FOR 3 DIRECTION AND STORE GRID COORDINATES
        IF (IREV.GE.2.OR.IWRIT.EQ.0) GO TO 750
        IF (IREV.EQ.1) THEN
        JREV = 15
        WRITE (12,1004) NX,NY
        ELSE
        WRITE (12,1003) NX,NY
        JREV = 8
        END IF
        YDIF  = -YN(JREV,JBO)
        DO 2224 J=1,NY
        DO 2224 I=1,NX
        YN(I,J)  = YN(I,J)+YDIF
 2224   CONTINUE
        RAD   = 4.777
        RAD1  = 4.503
        DO 790  J =1,NY
        DO 790  I =1,NX
  790   YNN(I,J)  = YN(I,J)/RAD
        DO 810 K = 1,15
        RAD = RAD1 +(K-1)*0.0686
        WRITE(12,1005) RAD,K
        DO 700 J=1,NY
        DO 700 I=1,NX
        X   = XN(I,J)
        Y   = RAD*SIN(YNN(I,J))
        Z   = RAD*COS(YNN(I,J))
        WRITE(12,1001) I,J,X,Y,Z
 700    CONTINUE
 810    CONTINUE
 750    CONTINUE
 1001   FORMAT(2I8,8F12.6)
 1002   FORMAT(8F12.6)
 1003   FORMAT(40X,10HROTOR GRID,5H NX =,I5,5H NY =,I5)
 1004   FORMAT(40X,11HSTATOR GRID,5H NX =,I5,5H NY =,I5)
 1005   FORMAT(//40X,8HRADIUS =,F12.6,4H K =,I5)
 10     FORMAT(2I8)
 40     FORMAT(2F12.6,2I8)
        STOP
        END
      SUBROUTINE  INTPL(MI,NI,SI,FI,M,N,S,F,FP,FPP,FPPP,MODE)
C     INTERPOLATION USING TAYLOR SERIES
C     ADDS CORRECTION FOR PIECEWISE CONSTANT FOURTH DERIVIATIVE
C     IF MODE GREATER THAN 0
      DIMENSION   SI(1),FI(1),S(1),F(1),FP(1),FPP(1),FPPP(1)
```

```
      K          = IABS(N  -M)
      K          = (N  -M)/K
      I          = M
      MIN        = MI
      NIN        = NI
      D          = S(N)  -S(M)
      IF (D*(SI(NI)  -SI(MI))) 11,13,13
   11 MIN        = NI
      NIN        = MI
   13 KI         = IABS(NIN  -MIN)
      IF (KI) 21,21,15
   15 KI         = (NIN  -MIN)/KI
   21 II         = MIN  -KI
      C          = 0.
      IF (MODE) 31,31,23
   23 C          = 1.
   31 II         = II  +KI
      SS         = SI(II)
   33 I          = I  +K
      IF (I  -N) 35,37,35
   35 IF (D*(S(I)  -SS)) 33,33,37
   37 J          = I
      I          = I  -K
      SS         = SS  -S(I)
      FPPPP      = C*(FPPP(J)  -FPPP(I))/(S(J)  -S(I))
      FF         = FPPP(I)  +.25*SS*FPPPP
      FF         = FPP(I)  +SS*FF/3.
      FF         = FP(I)  +.5*SS*FF
      FI(II)     = F(I)  +SS*FF
      IF (II  -NIN) 31,41,31
   41 RETURN
      END
      SUBROUTINE  SPLIF(M,N,S,F,FP,FPP,FPPP,KM,VM,KN,VN,MODE,FQM,IND)
C     SPLINE FIT
C     INTEGRAL PLACED IN FPPP IF MODE GREATER THAN 0
C     IND SET TO ZERO IF DATA ILLEGAL
      DIMENSION   S(1),F(1),FP(1),FPP(1),FPPP(1)
      IND        = 0
      K          = IABS(N  -M)
      IF (K  -1)81,81,1
    1 K          = (N  -M)/K
      I          = M
      J          = M  +K
      DS         = S(J)  -S(I)
      D          = DS
      IF (DS) 11,81,11
   11 DF         = (F(J)  -F(I))/DS
      IF (KM  -2) 12,13,14
   12 U          = .5
      V          = 3.*(DF  -VM)/DS
      GO TO 25
   13 U          = 0.
      V          = VM
      GO TO 25
```

```
14 U          = -1.
   V          = -DS*VM
   GO TO 25
21 I          = J
   J          = J  +K
   DS         = S(J)  -S(I)
   IF (D*DS)81,81,23
23 DF         = (F(J)  -F(I))/DS
   B          = 1./(DS  +DS  +U)
   U          = B*DS
   V          = B*(6.*DF  -V)
25 FP(I)      = U
   FPP(I)     = V
   U          = (2.  -U)*DS
   V          = 6.*DF  +DS*V
   IF (J  -N) 21,31,21
31 IF (KN  -2) 32,33,34
32 V          = (6.*VN  -V)/U
   GO TO 35
33 V          = VN
   GO TO 35
34 V          = (DS*VN  +FPP(I))/(1.  +FP(I))
35 B          = V
   D          = DS
41 DS         = S(J)  -S(I)
   U          = FPP(I)  -FP(I)*V
   FPPP(I)    = (V  -U)/DS
   FPP(I)     = U
   FP(I)      = (F(J)  -F(I))/DS  -DS*(V  +U  +U)/6.
   V          = U
   J          = I
   I          = I  -K
   IF (J  -M) 41,51,41
51 I          = N  -K
   FPPP(N)    = FPPP(I)
   FPP(N)     = B
   FP(N)      = DF  +D*(FPP(I)  +B  +B)/6.
   IND        = 1
   IF (MODE)81,81,61
61 FPPP(J)    = FQM
   V          = FPP(J)
71 I          = J
   J          = J  +K
   DS         = S(J)  -S(I)
   U          = FPP(J)
   FPPP(J)    = FPPP(I)  +.5*DS*(F(I)  +F(J)  -DS*DS*(U  +V)/12.)
   V          = U
   IF (J  -N)  71,81,71
81 RETURN
   END
```